BUILD A SMARTER AI-POWERED SPAM CLASSIFIER

```
import numpy as np

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, LSTM

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences


# Step 1: Data collection

# Assume you have a dataset with labeled spam and non-spam messages.

# Load and preprocess the data as needed.


# Step 2: Data preprocessing

# Convert text data into numerical format using techniques like TF-IDF or word embeddings.


# Example using TF-IDF

tfidf_vectorizer = TfidfVectorizer(max_features=5000)

X_tfidf = tfidf_vectorizer.fit_transform(X)  # X is a list of text messages


# Step 3: Split data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

# Step 4: Train a machine learning model (Naive Bayes)
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred_nb = nb_classifier.predict(X_test)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
report_nb = classification_report(y_test, y_pred_nb)

print(f'Naive Bayes Accuracy: {accuracy_nb}')
print(f'Classification Report:\n{report_nb}')

# Step 6: Train a deep learning model (LSTM)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X)
X_seq = tokenizer.texts_to_sequences(X)
X_padded = pad_sequences(X_seq)

model = Sequential([
    Embedding(input_dim=5000, output_dim=128, input_length=X_padded.shape[1]),
    LSTM(units=64),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_padded, y, epochs=5, batch_size=32, validation_split=0.2)
```

```
# Step 7: Evaluate the model

X_test_seq = tokenizer.texts_to_sequences(X_test)

X_test_padded = pad_sequences(X_test_seq, maxlen=X_padded.shape[1])



accuracy_lstm = model.evaluate(X_test_padded, y_test)[1]



    print(f'LSTM Accuracy: {accuracy_lstm}')
```